1984-10

Steps towards parsing of query sequences to a database / Proceedings of the First International Workshop on Expert Database Systems, October 1984

Steps towards parsing of query sequences to a database

Neil C. Rowe

Department of Computer Science

Code CS/Rp Naval Postgraduate School Monterey, CA 93943

Abstract

Sequences of queries to a database system can have structure. Recognizing this structure is a kind of "parsing", analogous to the parsing of sentences. We present two rather different approaches to recognition for exploitation. The first is a rule-based system that examines superficial aspects of a query sequence to postulate preferences between sets mentioned in the queries. The second is a deeper, but more limited model based on decision theory, which assigns utilities and suitability probabilities to individual set items, and attempts to explain set preferences on that basis. Both these methods have disadvantages, and their performance is difficult to analyze because of the fuzzy nature of the application, but it is hoped they can form the basis for more comprehensive man-machine interfaces.

This paper appeared in the Proceedings of the First International Workshop on Expert Database Systems, Kiawah Island, SC, October 1984, 786-797.

Introduction

Queries to a database query system usually occur in clusters. Though discourse-understanding issues such as anaphora and ellipsis are well-studied, the more general problem of inference of the user plan behind a connected sequence of queries remains elusive. This seems due to the difficulty of categorizing and recognizing the many different needs users have. After all, well-defined tasks tend to be better handled by batch processing; tasks with vague specifications and goals tend to work better with interactive query systems.

But we believe some progress can be made towards to goal of figuring out what users are up to, and adjusting system behavior accordingly. Some promising nonquantitative research has been done (Hobbs, 1978; Cohen, Perrault, and Allen, 1981; Reichman-Adar, 1984). Even small successes in understanding can have immediate payoffs in better management of previous query results, a crucial aspect of database operations. The usual method of throwing out the least-recently-used query results ignores a good deal of available information that often suggests better things to throw out. In addition, understanding query sequences pays off in more cooperative responses to queries, identification of previously unrecognizable user semantic errors, and support for new kinds of querying.

This paper synthesizes work summarized in (Rowe, 1984) with some new ideas. Many of the details (especially mathematical) omitted here may be found in that paper.

Some definitions

Analysis of a long query sequence as a unit is too hard. Our idea is thus to simplify the problem to the study of pairs of user queries, to see if we can recognize preference phenomena between the results of each query, including both differences in the resulting set compositions and attributes. We will identify "preferences" between such pairs (Luce, 1959), yes/no phenomena that can, however, be quantified by a certainty factor in the manner of rule-based systems such as MYCIN (Buchanan and Shortliffe, 1983). We will postphone details of the certainty factors to section 4.

By "query set" we will mean a set of some items represented in a database, items with the same data-type, and with some associated attributes of those items. We use "predictive power" to measure preference phenomena. That is, a user "prefers" one item in a database to another item if he is more likely to include that item (directly or indirectly) in future query sets. Preference between query sets, however, is more complicated and more "psychological" because it must take into account the sizes of the query sets as well as the total likelihood that some member of the set will be included in a future query set, for otherwise a set would always be preferred to its subset. In other words, we prefer query set i to query set j if

```
R sub i  - alpha { R sub i } / { N sub i }     >
R sub j - alpha { R sub j } / { N sub j }
```

where %R sub k% is the probability that some member of set i will be included in a future query set, %N sub k% the size of set k in items, and %alpha% a constant. (See section 5.3 for more details.)

Heuristics for query-output preference

We now suggest some heuristics that may be used to determine the preferences and their certainty factors, approximately in order of decreasing strength. We assume a relational database. We identify three basic kinds of tasks a user might be pursuing in using the database query system (see (Miller, 1969) for more discussion): (1) choosing something from among options for real-world action; (2) generating a report for management on some important part of the database; and (3) preparing a statistical analysis of significant phenomena in the data. We write user queries in standard font, system responses in italics.

Real-world connection

```
Which suppliers of widgets are located in California?

There are 33 names.

Send to the printer the address, location, and price of those
with the ten cheapest prices.
```

If a user employs some query set in the real world in preference to another, that usually means he prefers the former to the latter. The main difficulty is figuring out what happened in the real world. This depends on how policies decided with the help of the database are exhibited later in the data. If the database is the actual tool used to accomplish things -- as when an order can be made merely by adding an order record to an order relation -- one can merely search that relation to find out what happened. If the user's goal is only to prepare a report, then sending records or statistics on some query set off to a printer or special graphics device suggests that query set is preferred to another set not so treated.

Implementational "handles"

```
How many suppliers of widgets are located in California?

There are 33.

List the names of the ten nearest.

<listing>
```

If a specific real-world action is implemented from database information (as opposed to statistical analysis), "handles" on the data are often necessary. That means something that links the database records with real-world entities, like a name, identification number, or even some unique description. A query set with this information is preferable to one without.

Implementation preconditions

```
List the names of widget suppliers located in California.

<33 names>

Give their addresses and phone numbers.

<listing>
```

A related but distinct issue to the last heuristic is the preconditions that are often necessary for real-world implementation. Often just unique identification is enough, but certain unique identifiers make action much easier than others. You can usually phone a person knowing only their name, but knowing the phone number speeds things considerably.

Distinguishability

```
List the widget-like products supplied by California widget suppliers.

<65 listings, but "widget" the only product listed>

List the sizes of widgets supplied by California widget suppliers.

<listing with seven different size values>
```

If the user's goal is to choose something from the database, there has to be a basis for a choice. That is, there must be different values displayed for some attribute of some of the items in a query set. A set without such "distinguishability" (as the first set above) is less desirable than one that has it (as the second set above, which gives different sizes for different values).

Expletives

```
List the California widget suppliers and their addresses.
```

```
<listing of 33 names and addresses>

Great.  What are their prices on crates of widgets?

<listing of 33 prices>

Ugh.  What about prices on suppliers in other Western states?
```

A natural language query environment has some important advantages over a formal query language environment (e.g., SQL or QUEL): non-goal, thematic information can be exploited. Human information-seeking conversation contains many kinds of evaluative cues, and users could be encouraged to volunteer them to a database query system too. Many of these cues are one-word expletive tags on the beginning of sentences, and are easy to parse. Often their meaning is quite clear. "Good", "ok", "fine", "great", "swell", and "amazing" denote positive preferences to query results not so tagged. Similarly, "ugh", "argh", "oh no", "bad", "stop", and assorted profanity denote negative preferences. Some expletives are unclear, as "hmm", "wait", and "funny". Note that these cues occur *after* the query result, so they are found in a different place than evidence for other heuristics.

Repeated mention

```
What suppliers of widgets are in California?

<33 names>

What are their addresses and prices per crate?

<33 addresses and prices>
```

Repeated references to the same set suggest that set is preferred to others. Each repetition increases the likelihood.

Lateness of use

```
What suppliers of widgets are in California?

<33 names>

OK, bye.
```

People usually stop searching when they find what they are looking for. If the goal of a query session is to find something (as opposed to explore or browse), then the last query usually gives the information the user most prefers. Other query sets late in a session may be similar (the queries following them may just be "doublechecking"), so being late in a session should have weight too. This heuristic has similarities to the least-recent-used priority method, but it need not be so linear -- it usually doesn't matter much whether something was used 10 queries ago or 15, since neither result has been used in a long while and is not likely to be relevant.

Subsetting

```
What suppliers of widgets are in California?

<33 names>

Which have widgets under |20 a crate?

<16 names>

Which have widgets |20 to |30 a crate?

<12 names>
```

When subsets of a query set are subsequently taken, it suggests the set is more important than those not so treated. Each additional subset increases the preference of the set. Note the subset must occur after the set for this to be meaningful. The phenomenon is similar to that with repeated references to the same set, but not as strong in establishing preference.

Attribute exhaustion

```
Tell me everything you know about widgets from California suppliers.
```

If the database has supplied everything it has about a query set, a user is less likely to ask future questions about that set as opposed to some other. But the user might repeatedly ask for the same data when trying to prepare a table or a report for presentation.

Statistical interest

```
What are the widget suppliers in California?

<33 names>

What fraction of the widget suppliers are current manufacturing frobs?

<all but one>
```

If there is a statistical-analysis aspect to the user's task, any unexpected results make a set more interesting than those without such results. One can define "unexpected" as the degree to which counts and sums cannot be predicted from independence-assumption models and other linear models, or one can exploit detailed causal-relationship models (Blum, 1982).

Small query sets

```
What are the widget suppliers in California?

<33 names>

Which are within 10 miles?

<2 names>
```

If users are trying to choose a single object, they tend to narrow possibilities progressively. If so, the small (but nonempty) sets towards the end of the query session are preferable, when the most factors have been taken into account.

Putting the heuristics together

Certainty factors

We now suggest some reasonable quantitative rankings of the above heuristics, for the three types of user tasks mentioned. We assume the user's task is known beforehand. For these certainty factors, larger numbers mean greater certainty, and the numbers are scaled with 1.0 the largest (to suggest probabilities).

#	Heuristic name	Choice task	Report gen.	Stat. analysis
1	Real-world connection	1.0	1.0	1.0
2	Implementation handles	.9	.9	.9
3	Impl. preconditions	.8	na	na
4	Distinguishability	.8	.8	na
5	Expletives	.6	.6	.6
6	Repeated mention	.4	.5	.5
7	Lateness of use	.6	.6	.5
8	Subsetting	.3	.3	.4
9	Attribute exhaustion	.5	.1	.1
10	Statistical interest	na	.2	.7
11	Small query sets	.4	.1	.1

Combining certainty factors

We can evaluate any pair of query results in the output for a user query session, and assign a subset of the above certainty factors (chosen from the appropriate column) to the pair. To get a cumulative certainty factor, the much-used independence model of MYCIN (Buchanan and Shortliffe, 1983) seems reasonable:

```
CF sub total   =   1 - PI from i=1 to m left ( 1 - CF sub i right )
```

We can then build a directed graph with nodes representing the query results, and with numbers associated with each node representing the certainty factor in the preference of one result to another. In general, only a few of the possible connections may be made in the graph, so usually we do not have direct evidence of which of two specific sets is preferable. But we can assume that preference is transitive: if A is preferred to B, and B to C, A should be preferred to C. To compute the certainty factor for such implied preferences, we don't want to use an

independence assumption as we did in the formula above because clearly the preferences are not independent -- they refer to common nodes -- so instead we suggest the "conservative" approach taken by fuzzy logic and make the total preference the largest of the preferences in the path.

An example

Suppose we have the following query session. Again, the system in italics. We number queries and responses to more easily refer to them.

1. How many tankers are in the Mediterranean?
2. 37.
3. List the American ones.
4. Titanic, Bounty, Pequod, Lusitania, Pueblo, Mayaguez.
5. Give the tonnages for those more than 1000 feet.
6. None are that long.
7. Give the tonnages and positions for those over 500 feet.
8.

```
Ship    Tonnage Position
Bounty  14000   40N13E
Pequod  8000    45N5E
Pueblo  17000   43N18E
```

9. Good. What are the captain and radio call sign of the Pequod?
10. Ahab and WHL.
11. And who owns it?
12. Peleg Enterprises.

Assuming that the user is following a choice task:

Real-world connection (Heuristic 1) does not apply.

The query sets in queries 3, 5, 7, 9, and 11 are preferred to that of query 1 by the Implementation Handles Heuristic (Heuristic 2).

Query set 9 is preferred to 1, 3, 5, and 7 by the Implementational Preconditions Heuristic (Heuristic 3).

Sets 3, 5, 7, 9, and 11 are preferred to 1 by Distinguishability (Heuristic 4).

Set 7 is preferred to all others by the Expletives Heuristic (Heuristic 5).

Set 11 (= set 9) is preferred to 1, 3, 5, and 7 by Repeated Mention (Heuristic 6).

Set 11 (= set 9) is also preferred to 1, 3, 5, and 7 by Lateness of Use (Heuristic 7).

Heuristic 8 (Subsetting) applies to every set except 5, causing a preference of each set to its successors.

Heuristic 9 (Attribute Exhaustion) would apply to set 11 (= set 9) only if name, tonnage, position, captain, owner, and radio call sign were the only things known about ships. Let us assume this is not true.

Heuristic 10 (Statistical Interest) does not apply.

Set 11 (= set 9) is preferred to 1, 3, 5, and 7 by the Small Query Sets Heuristic (Heuristic 11).

We can create a 5 by 5 matrix representing the preference graph. The entries collect the certainty factors (if any) for the preference of that row to that column.

	set 1	set 3	set 5	set 7	set 9 (=11)
set 1	-	-	-	-	-
set 3	.9,.8,.3	-	-	-	-
set 5	.9,.8,.3	.3	-	-	-
set 7	.9,.8,.6,.3	.6,.3	.6,.3	-	.6
set 9 (=11)	.9,.8,.8,.4,.6,.3,.4	.8,.4,.6,.3,.4	.8,.4,.6,.3,.4	.8,.4,.6,.3,.4	-

Using the combination formula, the total certainty factors for pairs of query sets are as follows:

	set 1	set 3	set 5	set 7	set 9 (=11)
set 1	-	0	0	0	0
set 3	.960	-	0	0	0
set 5	.960	.300	-	0	0
set 7	.994	.720	.720	-	.6
set 9 (=11)	1.000	.980	.980	.980	-

More careful user modelling with decision theory

These certainty factors are crude, however, and only take into account superficial aspects of what a user is trying to do. They are somewhat robust, however, and can apply to a broad range of tasks. But if we are willing to narrow our focus somewhat, we can do better. One approach explored in detail in (Rowe, 1984) is to model certain choice tasks using detailed decision-theory models.

Utilities and suitabilities for choice tasks

When a transportation planner is choosing a vehicle to carry a load somewhere, many factors must be taken into account. Some have to do with the costs associated with alternatives, others with the availability and reliability of options. Following decision theory, we call the former utilities, and the latter probabilities -- though the latter are also a special kind of probability, which for lack of a better term we call suitabilities. As an example, in merchant shipping the utilities are the financial cost of loading a ship; the fuel, crew wages, and miscellaneous transit costs for a voyage; and the time delay in getting a cargo to its destination. Suitabilities are the ability of a ship to carry a particular kind of cargo; and the ability of a ship (due to its dimensions) to be serviced at a particular port.

We can sum up sub-utilities to get total utility %u sub j% for an option, and multiply sub-suitabilities (making a reasonable independence assumption) to get a total suitability %s sub j%. Then using a simple psychological model of how people make choices we have a formula for the probability %p sub i% of absolute preference of item i to all other items in a set of n items:

```
p sub i = {s sub i} / {1 - .5 s sub i}   PI from j=1 to N
left ( 1 - s sub j PHI left ( { u sub i - u sub j } / { sigma sub indiff }
right ) right )
```

where %PHI% is the integral of the unit normal curve about zero. Related formulas to this are discussed in (Luce, 1959).

As an example, consider four items with suitability-utility pairs (.8,20), (.5,15), (.4,10), (.2,15), and suppose %sigma sub indiff% is 5. Then:

```
p sub 1 = .29 , p sub 2 = .52 , p sub 3 = .87 , p sub 4 = .43
```

which we can normalize to:

```
p sub 1 = .14 , p sub 2 = .25 , p sub 3 = .41 , p sub 4 = .20
```

So the third item is the most preferred, with a probability around .41.

Slot filling

To provide some generality in our decision theory model, so we don't have to write separate formulae for every different transportation situation, we provide "slots" (in the sense of slots in frames) that are instantiated dynamically from the query sequence. These then force a choice among a class of similar utility and suitability assignments. Four common slots for choice tasks are:

1. Domain of discourse. These are restrictions mentioned repeatedly, that essentially limit nonegligible suitabilities to only certain categories. For example, the user asks repeatedly about subsets of American tankers.

2. Reference standard. These give a point against which objects are measured. For example, if the user asks for ships within 100 nautical miles of Naples, this suggests that Naples is the reference standard for location, and also suggests the user contemplates arriving there or departing from there.

3. Threshold values. These give criterion values suggesting exact quantities involved in real-world activities. For example, if the user asks for ships with more than 10000 ton capacity, it suggests he wants to transport 10000 tons.

4. Answer set size. These indicate the usual size of a query answer for a particular user, and suggest the degree to which the choice

problem is focussed (the %alpha% discussed below).

Evaluating sets

We now have a way to evaluate and compare sets: these probabilities for the items in the set. Following as before the formula for the probability union assuming independence, we can give a cumulative number for the desirability of at least one item in a set:

```
D = 1 - PI from i=1 to n left ( 1 - p sub i right )
```

But following the lead of information retrieval (Lancaster, 1979), this is misleading, for it would mean that larger sets would tend to be the most desirable. If a set is large, there may be many items with very low desirabilities, and this seems unfair. What we really want for total desirability is some weighted sum of the above D and D/n, the density of desirability in the set. The exact weighting %alpha% can be user-dependent -- see the discussion of user hierarchies below.

So to compare two sets, we just compute

```
E = D + { alpha D } / n
```

where D is given by the preceding formula; the set with the larger E is preferable.

Coordinating the two approaches to preferences

We have presented two rather different approaches to "parsing" of query sequences: one superficial but broadly applicable, and one deeper but complicated and requiring formalization of many subjective judgements. The obvious question is whether the two can be reconciled. We think so, on the basis of some preliminary experiments.

We applied the two methods to the same query sequences and compared results. Since the heuristic method is simpler, we used it as the standard, and adjusted parameters of the decision-theory approach until its preference results came out the same as often as possible. The parameters we adjusted were the weights on individual sub-utilities and sub-suitabilities (that is, their importance to the total utility or total suitability), though in a more comprehensive approach we could include other kinds of parameters like the %alpha% mentioned above which we assumed was zero.

We treated preferences as binary for this analysis -- that is, if the certainty was greater than a criterion we assumed a positive preference, else no preference. We then set up inequalities representing matchings of pairs of items, one drawn from each set, with the inequality sign in the direction of the preference. The paired items had to be distinct -- we removed common items from the two sets. This gives a large set of linear inequalities, most of which are redundant (derivable from others), so we eliminated the redundancies to get a small set of irredundant inequalities. We can also get inequalities from two other sources:

1. from past user behavior, suggesting reasonable bounds on utility and suitability values

2. from the query sequence directly, when a query set is a subset of the immediately previous query set. Since subsetting is evidence of preference, it suggests that the additional restrictions correspond to utilities or suitabilities that are better than those for items in the complement of the restriction. For instance, if the user next asks for the American ships in a set, it suggests that the nationality suitability for American ships is greater than the suitability for any other nationality.

"Solving" the inequalities

Our inequalities do not represent absolute criteria, only evidence. All we want is a representative point in the hyperregion defined by them. Thus we do not want to "solve" them per se, just find an answer most consistent with them. We can treat this as an optimization problem with "penalty functions" corresponding to the inequalities (Gill, Murray, and Wright, 1981) -- the fewer inequalities violated, the larger the value of the optimization function. Such a function can be expressed as follows:

```
PI from j=1 to N left ( { PHI ( DELTA sub j ) } over
{ sigma sub indiff } right )
```

where N is the number of inequalities, %sigma sub indiff% a measure of sensitivity to inequality violation, and %DELTA sub j% the difference of the right side from the left side of the jth inequality, where the inequality sign is turned if necessary to point right. And as before, %PHI% is the integral of the unit normal curve about zero.

Defaults from user hierarchies

Users will differ a good deal in parameters for the decision theoretic model. They will also differ to a lesser extent in the certainty factors they

would judge appropriate for the heuristic model. So user modelling is important for both approaches. But users form groups and subgroups based on people, time, and task. Multiple inheritance of quantitative parameters in the manner of (Rowe, 1982) can be used to provide starting defaults for the parameters, which can then be adjusted using the methods of the last section among others. Adjustments can then be averaged into the defaults for the groups to which the user belongs. To make this work, the user might be asked to critique default group assignments at the beginning of each session.

Applications

We can use preferences among query sets or query items for several purposes:

1. We can develop "knowledge-based temporaries management", intelligent deciding of which of the previous query results to save and which to throw away when space is needed.

2. We can prefetch data the user is likely to need, in the perhaps otherwise wasted time while he is examining one query result and deciding what to ask next.

3. With a decision-theory model of item preference, we can make suggestions to the user for what restrictions in a query to relax when a query set is empty or too small.

4. We can notice new classes of user errors. With a decision theory model, we can point out incorrect parts of a query expression, or necessary missing parts, that would lead to fetching of items with very low preference.

5. We can handle queries with vague or fuzzy restrictions, since the preference mathematics is probabilistic anyway.

Conclusion

It is difficult to evaluate man-machine interface innovations, and this area is no exception. We have proposed two approaches to a new and virtually uncharted area. We have tried to justify carefully the steps we have taken, but only detailed experimentation and further study with these approaches will provide the final judgement.

Acknowledgements

This work was partially supported by the Knowledge Base Management Systems project at Stanford University under contract #N00039-82-G-0250 from the Defense Advanced Research Projects Agency of the United States Department of Defense. It was also supported in part by the Foundation Research Program of the Naval Postgraduate School with funds provided by the Chief of Naval Research.

References

(Blum, 1982) Blum, R. L. Discovery, confirmation, and incorporation of causal relationships from a large time-oriented clinical database: the RX project. Computers and Biomedical Research, 15, (1982), 164-187.

(Buchanan and Shortliffe, 1983) Buchanan, B. G. and Shortliffe, E. H. Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley, 1983.

(Cohen, Perrault, and Allen, 1981) Cohen, P., Perrault, C. R., and Allen, J. Beyond question-answering. Bolt Beranek and Newman technical report 4644, May 1981.

(Gill, Murray, and Wright, 1981) Gill, P. E., Murray, W. and Wright, M. H. Practical optimization. New York: Academic Press, 1981.

(Hobbs, 1978) Hobbs, J. Why is discourse coherent? SRI International Artificial Intelligence Center technical note 176, November 1978.

(Lancaster, 1979) Lancaster, F. W. Information retrieval systems. New York: Wiley, 1979.

(Luce, 1959) Luce, R. D. Individual choice behavior. New York: Wiley, 1959.

(Miller, 1969) Miller, R. B. Archetypes in man-computer problem solving. IEEE Transactions on Man-Machine Systems, December 1969, MMS(10), 219-241.

(Reichman-Adar, 1984) Reichman-Adar, R. Extended person-machine interface. Artificial Intelligence, 22 (1984), 182-218.

(Rowe, 1982) Rowe, N. C. Inheritance of statistical properties. Proceedings of the National Conference, American Association for Artificial

Intelligence, Pittsburgh, Pennsylvania (August 1982), 221-224.

(Rowe, 1984) Rowe, N. C. Degrees of interest in a general database query system. International Journal of Man-Machine Studies, 20, 421-443 (May 1984).

Go to paper index